IEEE754 単精度-倍精度計算，多倍長浮動小数点計算を用いた混合精度反復改良法の性能評価

Performance Evaluation of Mixed-precision Iterative Refinement Method for Solving Linear Systems of Equations using IEEE754 Single-Double Precision or Multiple Precision Floating-point Arithmetic

幸谷智紀*

Tomonori KOUYA*

Abstract: Buttari et. al. propose the mixed-precision iterative refinement method using IEEE754 single and double precistion arithmetic for solving linear systems of equations. It is well-known that their proposed method can obtain high performance on computation environment on which the performance of single precision arithmetic is higher than double precison arithmetic. In this paper, we experiment how the original mixed precision iterative refinement method can perform on standard PC enviroment with IEEE single and double precision linear algebra computation libraries such as BNCpack, LAPACK and ATLAS. Furthermore, we show the result of performance evaluation and error estimation for extended one using multiple precision arithmetic such as MPFR/GMP.

1. 初めに

本稿では，Buttari らの提案した混合精度反復改良法 [2,7] を多倍長化し，数値解の精度評価と全体的な性能評価を行った結果について報告する。

反復改良法は 1970 年に Moler が提案したものである。n 次元方程式

$$\mathbf{f}(\mathbf{x}) = 0,\ \mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n \tag{1}$$

を解くため Newton 法を適用すると，その漸化式は

$$\mathbf{x}_{k+1} := \mathbf{x}_k - \left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}_k)\right]^{-1} \mathbf{f}(\mathbf{x}_k) \tag{2}$$

となる。ここで $[\partial \mathbf{f}(\mathbf{x}_k)/\partial \mathbf{x}]$ は Jacobi 行列である。

もし (1) が線型方程式

$$\mathbf{f}(\mathbf{x}) = A\mathbf{x} - \mathbf{b}$$

であれば，Jacobi 行列は定係数行列 A となるので，次のようなアルゴリズムで反復を進めることになる。

$$1.\ \mathbf{r}_k := \mathbf{b} - A\mathbf{x}_k \tag{3}$$

$$2.\ \text{Solve } A\mathbf{z}_k = \mathbf{r}_k \text{ for } \mathbf{z}_k \tag{4}$$

$$3.\ \mathbf{x}_{k+1} := \mathbf{x}_k + \mathbf{z}_k \tag{5}$$

これが連立一次方程式向けの反復改良法である。初期値 $\mathbf{x}_0$ が最初から要求される精度に達していれば反復する必要はないが，有限桁の浮動小数点数点演算を使用すると丸め誤差の影響で残差 $\mathbf{r}_k$ がゼロにならないため，これを最小化するように複数回の反復が行われる。そのため，近似解 $\mathbf{x}_k$ の精度を上げるためには，残差の計算は高精度で行う必要がある。

Buttari らは，A の条件数 $\kappa(A) = \|A\|\|A^{-1}\|$ が，使用する浮動小数点数の精度に比してあまり大きくない場合，1 の残差計算の精度より 2 の計算精度を低くしても，収束するための十分条件が成立する (縮小写像になる) ことを示し，1 を IEEE754 倍精度計算 (以下，倍精度と略記)，2 を IEEE754 単精度計算 (以下，単精度) することで全て倍精度計算で実行した時よりも計算効率が上がることをベンチマークテストで示した。しかしこの計算効率の向上は，単精度計算が倍精度計算よりも格段に高速に実行できる環境でなければなしえないものである。例えば Cell Broadband Engine のような単精度計算が非常に高速な CPU や，Cache ヒット率最適化や SIMD 命令によって最適化された線型計算ライブラリを用いた環境がそれにあたる。

本稿では，まず Buttari らの提案する混合精度反復改良法の概要を示し，そのまま多倍長計算でも使用できることを確認する。次に実際の PC 環境で，倍精度計算でも反復改良法の収束が保証される良条件問題と，多倍長計算でなければ保証されない悪条件問題に対してベンチマークテストを行い，近似解の精度とアルゴリズムの効率性がどの程度得られるのかを示す。最後に今後の研究課題を示す。

2. 混合精度反復改良法の概要

解くべき n 次元連立一次方程式を

$$A\mathbf{x} = \mathbf{b}$$
$$A \in \mathbb{R}^{n\times n},\ \mathbf{x} \in \mathbb{R}^n,\ \mathbf{b} \in \mathbb{R}^n \tag{6}$$

とし，係数行列 A は正則行列とする。本稿では，(6) における $A, \mathbf{b}$ の全成分は任意の精度を持つように設定できるものとする。

Buttari らは，混合精度反復改良法は，後述する収束条件を満足すれば，通常の連立一次方程式の解法を全て L 桁で計算した時に得られる近似解の精度と同程度の精度が得られるとしている。この時，計算の効率を上げるために，(4) の計算は $S(< L)$ 桁で実行する必要がある。この部分で使用する解法は，安定しているアルゴリズムが望ましいとしており，具体的には GMRES 法や直接法を挙げている。今回は部分ピボット選択を用いた LU 分解による直接解法を使用する。この時，(4) は，予め A を LU 分解しておくと

$$(PLU)\mathbf{z}_k = \mathbf{r}_k$$

2009 年 3 月 3 日 受理

*総合情報学部コンピュータシステム学科

となる。当然反復の前に $A = PLU$ として分解しておき (P は部分ピボット選択による行の入れ替えを表現する行列)，反復過程では前進・後退代入のみ行う。

以上をアルゴリズムの形でまとめると，(3)〜(5) は以下のようになる。ここで，$A^{[S]}$, $\mathbf{b}^{[L]}$ はそれぞれ S 桁，L 桁の浮動小数点数で表現した行列・ベクトルを意味する。

1. $A^{[L]} := A$, $A^{[S]} := A^{[L]}$, $\mathbf{b}^{[L]} := \mathbf{b}$, $\mathbf{b}^{[S]} := \mathbf{b}^{[L]}$
2. $A^{[S]} := P^{[S]}L^{[S]}U^{[S]}$
3. Solve $(P^{[S]}L^{[S]}U^{[S]})\mathbf{x}_0^{[S]} = \mathbf{b}^{[S]}$ for $\mathbf{x}_0^{[S]}$
4. $\mathbf{x}_0^{[L]} := \mathbf{x}_0^{[S]}$
5. For $k = 0, 1, 2, \ldots$
   (a) $\mathbf{r}_k^{[L]} := \mathbf{b}^{[L]} - A^{[L]}\mathbf{x}_k^{[L]}$
   (b) $\mathbf{r}_k^{[S]} := \mathbf{r}_k^{[L]}$
   (c) Solve $(P^{[S]}L^{[S]}U^{[S]})\mathbf{z}_k^{[S]} = \mathbf{r}_k^{[S]}$ for $\mathbf{z}_k^{[S]}$
   (d) $\mathbf{z}_k^{[L]} := \mathbf{z}_k^{[S]}$
   (e) Exit if $\|\mathbf{r}_k^{[L]}\|_2 \leq \sqrt{n}\,\varepsilon_R\,\|A^{[L]}\|_F\|\mathbf{x}_k^{[L]}\|_2 + \varepsilon_A$
   (f) $\mathbf{x}_{k+1}^{[L]} := \mathbf{x}_k^{[L]} + \mathbf{z}_k^{[L]}$

この S-L 桁混合精度反復改良法が収束するための条件は，次のようになる。

S, L 桁計算時のマシンイプシロンをそれぞれ $\varepsilon_S, \varepsilon_L$ と表現すると，まず L 桁計算する (3) は

$$\begin{aligned} &\mathbf{r}_k = \mathbf{b} - A\mathbf{x}_k + \mathbf{e}_k \\ &\quad \text{ここで } \|\mathbf{e}_k\| \leq \varphi_1(n)\varepsilon_L\,(\|A\| \cdot \|\mathbf{x}_k\| + \|\mathbf{b}\|) \end{aligned} \tag{7}$$

と誤差 $\mathbf{e}_k$ を含めて表現でき，同様に (5) は

$$\begin{aligned} &\mathbf{x}_k = \mathbf{x}_k + \mathbf{z}_k + \mathbf{f}_k \\ &\quad \text{ここで } \|\mathbf{f}_k\| \leq \varphi_2(n)\varepsilon_L\,(\|\mathbf{x}_k\| + \|\mathbf{z}_k\|) \end{aligned} \tag{8}$$

と表現できる。更に (4) も

$$\begin{aligned} &(A + H_k)\mathbf{z}_k = \mathbf{r}_k \\ &\quad \text{ここで } \|H_k\| \leq \phi(n)\varepsilon_S\|A\| \end{aligned} \tag{9}$$

と表現できる [4]。

この時，$\alpha_F, \beta_F \in \mathbb{R}$ を

$$\begin{aligned} \alpha_F &= \frac{\phi(n)\kappa(A)\varepsilon_S}{1 - \phi(n)\kappa(A)\varepsilon_S} + 2\varphi_1(n)\kappa(A)\varepsilon_L + \varphi_2(n)\varepsilon_L \\ &\quad + 2(\varphi_1(n)\varepsilon_L)\varphi_2(n)\kappa(A)\varepsilon_L \\ &= \psi_F(n)\kappa(A)\varepsilon_S \end{aligned} \tag{10}$$

$$\begin{aligned} \beta_F &= 4\varphi_1(n)\kappa(A)\varepsilon_L + \varphi_2(n)\varepsilon_L + 4(1 + \varphi_1(n)\varepsilon_L)\varphi_2(n)\kappa(A)\varepsilon_L \\ &= \rho_F(n)\kappa(A)\varepsilon_L \end{aligned} \tag{11}$$

とおく。もし

$$\frac{\rho_F(n)\kappa(A)\varepsilon_S}{1 - \psi_F(n)\kappa(A)\varepsilon_S} < 1 \text{ かつ } \alpha_F < 1 \tag{12}$$

であれば

$$\lim_{k\to\infty} \|\mathbf{x} - \mathbf{x}_k\| \leq \frac{\beta_F}{1 - \alpha_F}\|\mathbf{x}\| \tag{13}$$

となり，ノルム相対誤差が $\beta_F/(1 - \alpha_F)$ 程度まで小さくなることが期待できる [2]。

以上の収束条件より，S-L 桁混合精度反復改良法が収束するためには，

$$\kappa(A)\varepsilon_S << 1 \tag{14}$$

でなければならないことが分かる。つまり，条件数 $\kappa(A)$ が大きければ，それに応じて S を大きく取れば良いことになるが，計算速度の向上は見込めなくなる。条件数が小さければ相応に S を小さくすることもできるが，そもそも L 桁も必要な計算なのかという疑問が湧いてくる。従って，S-L 桁混合精度反復改良法が有効なのは

- L 桁の精度が必要で，$\varepsilon_S^{-1} > \kappa(A)$ である時
- S, L が固定されており，S 桁計算が十分に L 桁計算より高速である環境にある時

に限られることが分かる (Fig.1)。

3. 数値実験

以上述べてきたように，混合精度反復改良法が有効である条件は問題の性質（特に条件数）と計算環境に依存する。Buttari らの数値実験は，単精度計算が倍精度計算より高速であるようなハードウェア・ソフトウェア環境の上で行われているのも，逆にそうでない環境では性能が発揮されないためと思われるが，実際どの程度パフォーマンスが悪化するかは不明である。また，扱う問題の条件数も明示されていないことから，条件数が良い問題と悪い問題ではどの程度パフォーマンスに差が出てくるのかも不明である。

そこで我々はまず単精度でも 10 進 2 桁以上の精度が得られる良条件の問題を生成し，一般的な PC 上において，倍精度計算ではどのようなどのようなソフトウェア環境であれば，混合精度反復改良法の性能を発揮させることができるかを，数値実験により明らかにする。更に，多倍長計算環境ではどの程度の性能を発揮できるかも調査する。

以上の結果を確認するため，非対称悪条件行列である Lotkin 行列を用いて，悪条件問題に対して多倍長計算を用いた反復改良法がどの程度有効であるかも調査を行う。

数値実験を行った計算環境は下記の通りである。

CPU AMD Athlon64X2 3800+

RAM 4GB

OS CentOS 5.2 x86_64

C compiler GCC 4.1.2

Multiple Precision Library MPFR 2.3.2[10]/GMP 4.2.1[1] + BNC-pack 0.7b[5]

Linear Alegebra Computation Library LAPACK 3.2[8], ATLAS 3.8.3[11]

3.1 倍精度計算のベンチマークテスト

条件数を固定して設定できるよう，一様乱数を用いて生成した正則行列 X と逆行列 X^{-1}，対角行列 $D = \mathrm{diag}(n, n-1, \ldots, 1)$ を用いて

$$A = XDX^{-1} \tag{15}$$

として密行列 A を作成した。これにより，常に $\kappa_2(A) = n$ という，次元数がさほど大きくなければ良条件の行列となる。また解ベクトル $\mathbf{x}$ は

$$\mathbf{x} = [1\ 2\ \ldots\ n]^T \tag{16}$$

とし，L 桁に正しく丸められた A 及び $\mathbf{b} = A\mathbf{x}$ を生成してテスト問題を作成している。メモリ容量の制約から，次元数 n は 4096 までにしてある。この問題では，全ての次元数・精度においても反復改良法は収束し，反復回数は 2〜3 に留まっている。

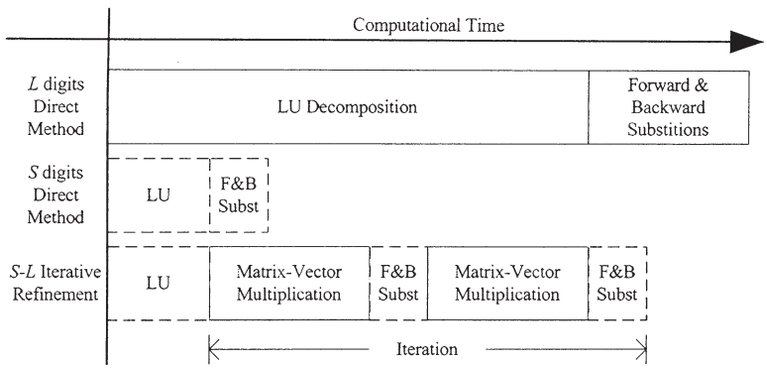

Fig. 1: 混合精度反復法の計算時間の構成

まず，Buttari らの提案した IEEE754 単精度・倍精度計算を用いた反復改良法の計算時間を Table 1 に示す。比較のため，単精度計算，倍精度計算での LU 分解法 (部分ピボット選択 LU 分解+前進・後退代入) の計算時間 LU_P(L), LU_P(S) も併せて提示してある。

全体として，単純な 3 重ループ+関数呼び出しによる行列インデックス計算を使用している BNCpack の計算時間が突出して計算時間を要していることが分かる。それに対して，一次元配列＋ブロック化を行っている LAPACK の計算時間は約 1/20〜7/100 に留まっている。ATLAS は LAPACK の主要関数を更に SIMD 命令・キャッシュの最適化を用いて高速化したものとされているが，この環境ではそれほど効果はなく，LAPACK より劣るケースも見受けられる。

前述したように，反復改良法の計算時間を決定するのは，単精度と倍精度計算時間の差が大きいときに限られる (Fig.1)。LU_P(L) が倍精度の LU 分解法の計算時間，LU_P(S) が単精度計算の LU 分解法の計算時間なので，この差が大きいほど反復改良法の計算時間が LU_P(L) より小さくなることが期待できる。このケースでは BNCpack と LAPACK がほぼ同様の計算時間となっており，ATLAS が次元数が大きくなるにつれて約 1.3 倍までその差が開いている。これは恐らく，BNCpack, LAPACK が，CPU の内部では拡張倍精度で計算しているのに対し，ATLAS は単精度向け，倍精度向けの SIMD 計算命令を使用しているためと思われる。以上の事由により，反復改良法の計算時間は，2048 次元以上の ATLAS 使用時のみ改善がみられる。BNCpack では LU_P(L) と殆ど同じであり，LAPACK では逆に小さくなっている。

得られた反復改良法と倍精度 LU 分解法の近似解の精度と，Table 1 に示した計算時間から算出した反復改良法の計算時間短縮比 (= S-L Iter.Ref./LU_P(L)) を Fig.2 に示す。近似解の精度は，同じ解法であればどのライブラリを用いてもその差は 10 進 1 桁以内に収まっている。反復改良法の精度は LU 分解法に比べて 10 進 2 桁程度悪化していることが分かる。しかし単精度計算の精度に比較するとほぼ倍の精度が得られており，ATLAS のように単精度計算の速度が高速なライブラリを用いれば計算時間面でもメリットがあることが分かる。

3.2 多倍長計算のベンチマークテスト

次に，多倍長計算を用いた場合の計算時間および精度の比較を行う。多倍長計算には前述したように BNCpack + MPFR/GMP を用いる。計算精度の指定は 10 進桁数 L を与え，$S = L/2$ として自動的に行う。メモリ容量の制限から，次元数は 1024, 計算精度は最大で 10 進 2000 桁までとした。

ソフトウェア実装による多倍長計算は計算精度に強く依存して計算時間が決まるため，ユーザの要求精度が条件数に比して大きい場合は混合精度反復改良法のメリットが生かせると予想される。S/L が小さいほど性能向上が図れることになるが，実際に適用する時には，未知の問題の場合，あらかじめ A を LU 分解して条件数の推定 [9] をすることを考えると，あまり小さな S を指定すると，LU 分解をやり直す必要が出てくる。その場合は反復改良法を使用するメリットは全くなくなるため，S はあまり小さくしない方が望ましい。今回は $S = L/2$ と固定したのはそうした事由による。

3.3 良条件テスト問題の場合

まず良条件の行列 (15) を係数行列 A，解ベクトル $\mathbf{x}$ も (16) とし，正しく指定精度で丸められた $\mathbf{b} = A\mathbf{x}$ を求めて作成した問題を用いてベンチマークテストを行う。反復改良法の反復回数はすべて 2 回で収まっている。計算時間の一覧表を Table 2 に示す。

LU_P(L) および LU_P(S) の計算時間の差が約 1.2〜2.0 倍出てくるので，どの計算精度においても LU_P(S) 以下の計算時間で実行できている。この時，計算時間の実行効率は計算精度が大きいほど上がり，最大 2 倍の効率アップとなっている (Fig.3)。

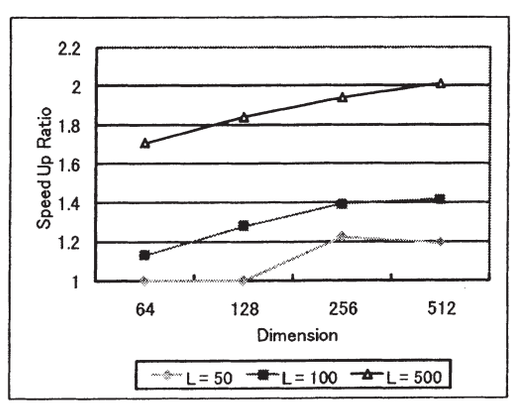

Fig. 3: 多倍長計算環境における計算時間短縮比 (良条件問題)

解の精度も IEEE 倍精度計算同様，反復改良法の方がどの計算精度においても 10 進 2 桁程度悪くなっているが，ほぼ同程度の精度は得られていると見てよい (Fig.4)。

Table 1: IEEE754 単精度 ($S = 10^{-6}$)・倍精度計算 ($L = 10^{-15}$) の計算時間 (単位 : 秒)

	S-L Iter.Ref.			LU_P(L)			LU_P(S)		
Dimension	BNCpack	LAPACK	ATLAS	BNCpack	LAPACK	ATLAS	BNCpack	LAPACK	ATLAS
512	2.83	0.14	0.05	2.82	0.1	0.05	2.83	0.14	0.05
1024	22.51	0.56	0.33	22.65	0.39	0.29	22.51	0.56	0.33
2048	184.26	2.23	1.83	185.56	1.54	2.06	184.26	2.23	1.83
4096	1422.95	10.11	11.61	1440.31	6.91	15.16	1422.95	9.73	11.61

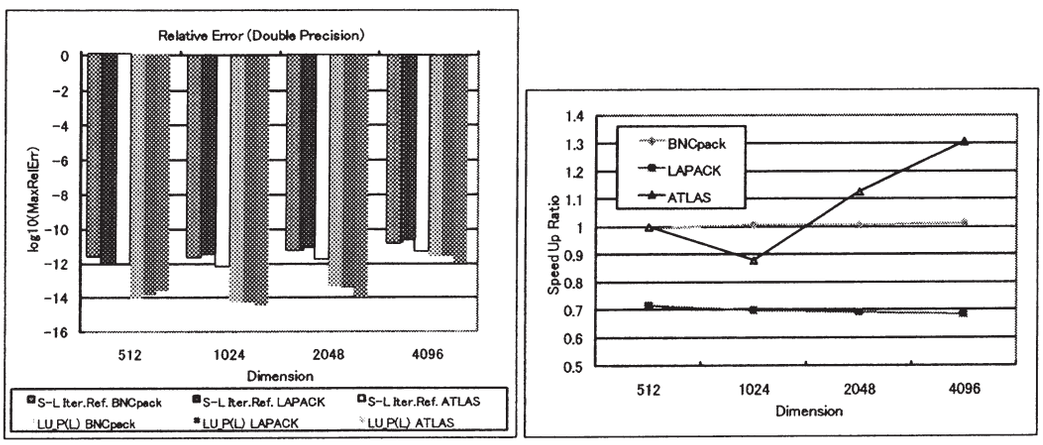

Fig. 2: 倍精度計算環境における最大相対誤差誤差 (左) と計算時間短縮比 (右)

Table 2: 多倍長計算時の計算時間 (単位:秒)

	$L = 50, S = 25$			$L = 100, S = 50$			$L = 500, S = 250$		
Dimension	Iter.Ref.	LU_P(S)	LU_P(L)	Iter.Ref.	LU_P(S)	LU_P(L)	Iter.Ref.	LU_P(S)	LU_P(L)
128	0.18	0.15	0.18	0.23	0.18	0.26	0.69	0.58	1.18
256	1.4	1.25	1.4	1.75	1.55	2.24	5.21	4.63	9.59
512	10.12	9.59	12.38	12.62	11.83	17.57	39.27	37.45	76.19
1024	78.89	74.17	94.22	99.17	91.52	140.57	301.55	292.05	606.7

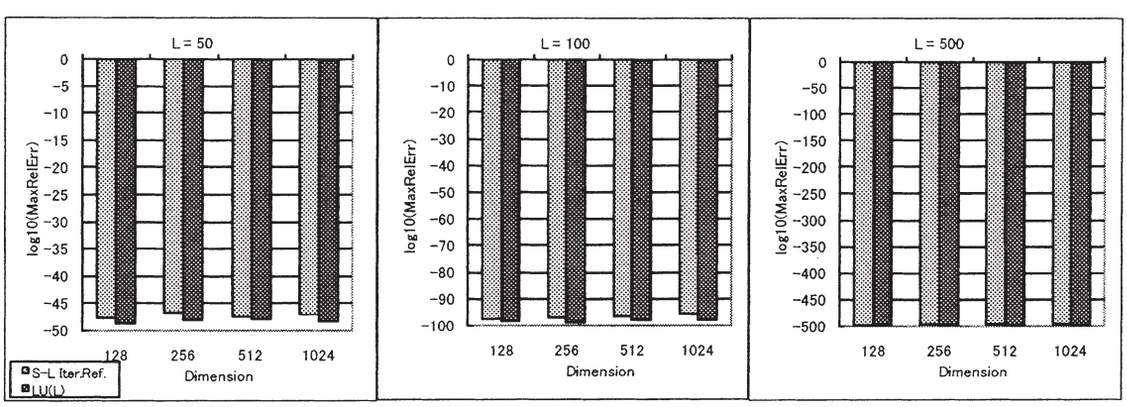

Fig. 4: 多倍長計算環境における最大相対誤差

3.4 悪条件問題の例

悪条件問題の例として，Hilbert 行列の第一行目を全て 1 にした Lotkin 行列 (17) を係数行列 A として使用する。解ベクトルは (16) として，定数ベクトルは $\mathbf{b} = A\mathbf{x}$ として指定精度で正しく丸めて与えた。

$$A = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1/2 & 1/3 & \cdots & 1/(n+1) \\ \vdots & \vdots & & \vdots \\ 1/n & 1/(n+1) & \cdots & 1/(2n-1) \end{bmatrix} \tag{17}$$

Lotkin 行列の条件数は Hilbert 行列と同じ程度に大きくなるが，計算精度より条件数が大きくなると，実際に計算精度 L で打ち切られた行列 $A^{[L]}$ の条件数は真の条件数より小さくなる (Table 3)。この例では 512 次元，10 進 500 桁の場合がそれに当たる。

Table 3: Lotkin 行列の条件数

	$\log_{10}(\kappa_1(A^{[L]}))$		
Dimension	$L = 500$	$L = 1000$	$L = 2000$
64	96.0	96.0	96.0
128	193.9	193.9	193.9
256	389.8	389.8	389.8
512	506.0	781.6	781.6

従って，500 桁計算では 128 次元まで，1000 桁計算では 256 次元まで，2000 桁計算でようやく 512 次元まで，混合精度反復改良法は収束するようになり，解の精度は Fig.5 に示したようになり，収束した場合は LU_P(L) と同程度の精度が得られていることが分かる。しかし，1000 桁，512 元問題のように，LU_P(L) が 200 桁程度の精度が得られる場合でも，800 桁程度の損失桁があると，$S = 500$ では収束条件を満たさなくなる。このような時は，例えば $S > 800 > L/2$ とすれば収束する可能性も出てくるが，そうなると計算時間短縮比は減る。

反復改良法が収束する計算精度・次元数の計算時間短縮比を Fig.6 に示す。

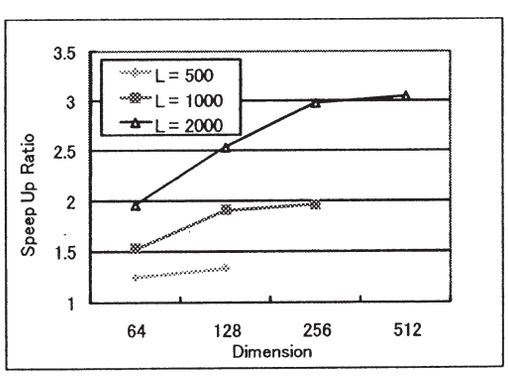

Fig. 6: 多倍長計算環境における計算時間短縮比 (Lotkin 行列)

良条件の問題同様 (Fig.3)，計算精度が大きくなるほど向上比は上がっており，LU_P(L) に比べて最大 1/3 の計算時間にすることができる。

4. 結論と今後の課題

以上の数値実験により，S-L 桁混合精度反復改良法は，収束条件を満足するときには $L-2$ 桁程度の精度が得られることが判明した。また IEEE754 単精度・倍精度計算の時のように，S 桁計算，L 桁計算の差が少ない時には計算時間の短縮にはつながらないケースが見られることも，多倍長計算の場合はより短縮効果が大きいことも判明した。

この結果より，今後の課題は次の三つが挙げられる。

4.1 他の反復法への適用

今回適用した直接解法は，一般の疎行列を係数行列とする問題の場合には計算量と使用メモリの節約が難しいことが知られており，そのような場合は (定常・非定常) 反復法を用いるのが普通である。混合精度計算を反復法に用いる提案は既に小武守ら [12] によってなされており，四倍精度と倍精度計算との組み合わせによって収束率と計算時間の向上が可能であることを数値実験によって示されているが，実装依存なアルゴリズムを部分的に使用しているため，使用する任意精度計算プログラムや反復法に制限が発生する。Buttari らの提案する手法をそのまま Krylov 部分空間法に適用した結果は，2008 年に Buttari らが別の論文 [3] で示しており，単精度で収束可能な問題に対しては，高速な疎行列用ライブラリを用いることで有効であることが確認されている。従って，多倍長計算環境においても同様の効果が期待できるため，今後数値実験によってその効果を確認していきたい。

4.2 古典的誤差評価法の適用

条件数の推定法としては，LU 分解を用いる方法がスタンダードになっている [9]。今回実験を行った LU 分解を用いる混合精度反復改良法においては，これをあらかじめ実行して $\kappa(A)$ を推定し，それを用いて収束条件 (12) を十分満たしていれば反復改良法を用い，そうでなければ改めて $L = U + \alpha$ として LU 分解をやり直して直接法を用いるかを判断することが可能になる。

従って，ユーザの要求精度を満足する解を得るための古典的誤差評価法 [6] にこのアルゴリズムを組み込むことで，高速な計算が可能になると考えられる。この考えを取り入れた任意精度計算プログラムを実装し，精度と計算時間の観点から性能評価を行いたい。

4.3 非線型方程式解法の内部反復法への応用

多次元非線型方程式を解くためのアルゴリズムとしては Newton 法が一番ポピュラーである。Newton 法の内部では Jacobi 行列を係数行列とする連立一次方程式を繰り返し解く必要があり，これを内部反復と呼ぶ。この内部反復に混合精度反復改良法を組み込み，精度を保ちつつ Newton 法の計算時間が減らせるかどうかの実験を行いたい。もしこれがうまくいくようであれば，陰的 Runge-Kutta 法の高速化にも使用することも考えている。

謝辞

本稿で述べた混合精度反復改良法の基本的なアルゴリズムの理解と，その実装は神尾亮介君の助力を得て行われた。教員の独断に基づく研究課題に対して真摯に取り組んだその姿勢に対して感謝申し上げる。

最後に，いつも真摯にご指導して頂いている元・日本大学永坂秀子先生に厚く御礼申し上げる。

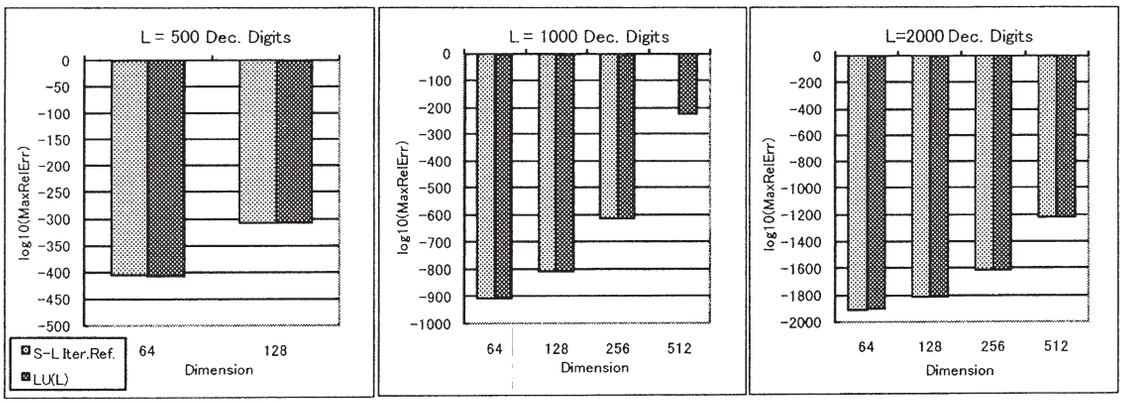

Fig. 5: 最大相対誤差誤差 (Lotkin 行列)

参考文献

1) Swox AB. GNU MP. http://gmplib.org/.
2) A.Buttari, J.Dogarra, Julie Langou, Julien Langou, P.Luszczek, and J.Karzak. Mixed precision iterative refinement techniques for the solution of dense linear system. *The International Journal of High Performance Computing Applications*, Vol. 21, No. 4, pp. 457–466, 2007.
3) A.Buttari, J.Dongarra, J.Kurzakand P.Luszczek, and S.Tomov. Using mixed precision for sparse matrix computations to enhance the performance while achieving 64-bit accuracy. *ACM Trans. Math. Softw.*, Vol. 34, No. 4, pp. 1–22, 2008.
4) G.W.Stewart. *Matrix Algorithms Volume I: Basic Decompositions*. SIAM, 1998.
5) Tomonori Kouya. BNCpack. http://na-inet.jp/na/bnc/.
6) 幸谷智紀. 実用的な古典的誤差評価法の提案と gauss 型積分公式の分点計算への応用について. 情報処理学会論文誌, Vol. 48, No. SIG18(ACS20), pp. 1 – 11, 2007.
7) Julie Langou, Julien Langou, Piotr Luszczek, Jakub Kurzak, Alfredo Buttari, and Jack J. Dongarra. Exploiting the performance of 32 bit floating point arithmetic in obtaining 64 bit accuracy (revisiting iterative refinement for linear systems). Technical Report 175, LAPACK Working Note, June 2006.
8) LAPACK. http://www.netlib.org/lapack/.
9) 松尾宇泰, 杉原正顯, 森正武. 行列の条件数の推定方法の数的評価. 日本応用数理学会論文誌, Vol. 7, No. 3, pp. 307 – 319, 1997.
10) MPFR Project. The MPFR library. http://www.mpfr.org/.
11) ATLAS: Automatically Tuned Linear Algebra Software. http://math-atlas.sourceforge.net/.
12) 小武守恒, 藤井昭宏, 長谷川秀彦, 西田晃. 倍精度と 4 倍精度の混合型反復法の提案. ハイパフォーマンスコンピューティングと計算科学シンポジウム (HPCS2007) 論文集, Vol. 2007, pp. 9–16, 2007.